TD1 : File

2023-2024

Dans ce TD, nous allons programmer une File. Une File est une suite d'éléments dans laquelle nous pouvons extraire l'élément placé en tête de la file et ajouter un élément en fin de file. Dans les deux premières parties de ce TD, nous allons implémenter une file dont la spécification est la suivante :

```java
public interface File {
    /**
     * @return vrai si la file est vide
     */
    public boolean estVide();

    /**
     * @return vrai si la file est pleine
     */
    public boolean estPleine();

    /**
     * @return le nombre d'elements presents dans la file
     */
    public int getTaille();

    /**
     * @param x: valeur a ajouter en fin de file
     */
    public void ajouter(float x);

    /**
     * supprimer l'element de tete de file.
     */
    public void supprimer();

    /**
     * @return valeur de l'element en tete de file
     */
    public float getTete();
}
```

1 Implémentation d'une file de réels à l'aide d'un tableau

Nous allons implémenter une file de réels en utilisant un tableau pour stocker les éléments. Le mode de fonctionnement sera le suivant : lorsqu'un élément est ajouté à la file, ce dernier est stocké dans la première case libre (case non utilisée pour stocker un élément de la file) du tableau. Lorsqu'un élément est extrait de la file, tous les éléments du tableau sont décalés de manière à ce que la case d'indice 0 du tableau contienne la nouvelle tête de file.
Exemple (le symbole / désigne une case non utilisée du tableau – sa valeur est donc indéfinie) :

État du tableau après quelques ajouts et suppression	5.5	1.3	15.2	1.5	/	/	/
État du tableau après l'ajout de la valeur 20.2 dans la file	5.5	1.3	15.2	1.5	20.2	/	/
État du tableau après le retrait de la valeur située en tête de file	1.3	15.2	1.5	20.2	/	/	/

1) Programmez la classe **FileTableau** qui implémente l'interface **File** selon le principe décrit ci-dessus.

2) Écrivez un programme *main* pour tester manuellement votre implémentation. Ce programme devra demander à l'utilisateur de saisir une suite de nombres à insérer dans la file. La saisie s'arrêtera lorsque la file sera pleine ou lorsque l'utilisateur aura saisi un nombre négatif. Une fois la saisie effectuée, le programme affichera le contenu de la file à l'écran.

3) Évaluez le coût des opérations de cette implémentation.

2 Implémentation sans décalage

1) Proposez et programmez une deuxième implémentation qui n'effectue aucun décalage d'élément.

2) Quelle(s) modification(s) faut-il apporter au programme de test de la question précédente pour pouvoir tester cette nouvelle implémentation de **File** ?

3 Une file générique

1) Que peut-on rendre générique dans la spécification **File** ?

2) Changez la spécification du type **File** pour la rendre générique.

3) Changez la deuxième implémentation de manière à la rendre générique.

4) Changez le programme de test de manière à refléter ces modifications.

Annexe : spécification du TA générique Array

```java
/**
 * Array : tableau de capacite fixe dont les elements sont d'un type T
 */
public interface Array<T> {
    /**
     * Initialiser une nouvelle instance de Array
     * @param capacite : capacite (nombre de "cases") du tableau
     */
    public Array(int capacite);

     /**
      * @return nombre de "cases" du tableau
      */
    public int length();

    /**
     * @param i : indice de l'element a consulter
     * @return valeur de l'element d'indice i
     */
    public T get(int i);

    /**
     * @param i : indice de l'element a modifier
     * @param v: nouvelle valeur de l'element d'indice i
     */
    public void set(int i, T v);
}
```